SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001-2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST,
COIMBATORE – 641 022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

20CS281– CLOUD COMPUTING LABORATORY

LAB RECORD

ACADEMIC YEAR: 2022-2023

BATCH: 2020-2024

APRIL 2023

SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of works done by Mr./Ms. _____ in 20CS281 - CLOUD COMPUTING LABORATORY of this Institution for VI Semester during the Academic Year 2022 – 2023.

Faculty In-Charge
Mrs. M. KARTHIGHA AP (Sr.G)/CSE

HOD – CSE
Dr. A. GRACE SELVARANI, Prof/Head

Date:

Register Number: _____

Submitted for the VI Semester B.E.-CSE Practical Examination held on _____ during the Academic Year 2022 – 2023.

Internal Examiner

Subject Expert

INDEX

EX.NO: 1	DEPLOY A WEBSITE IN A CLOUD
DATE:	

Aim:

To create a website and deploy it in a cloud.

Procedure:

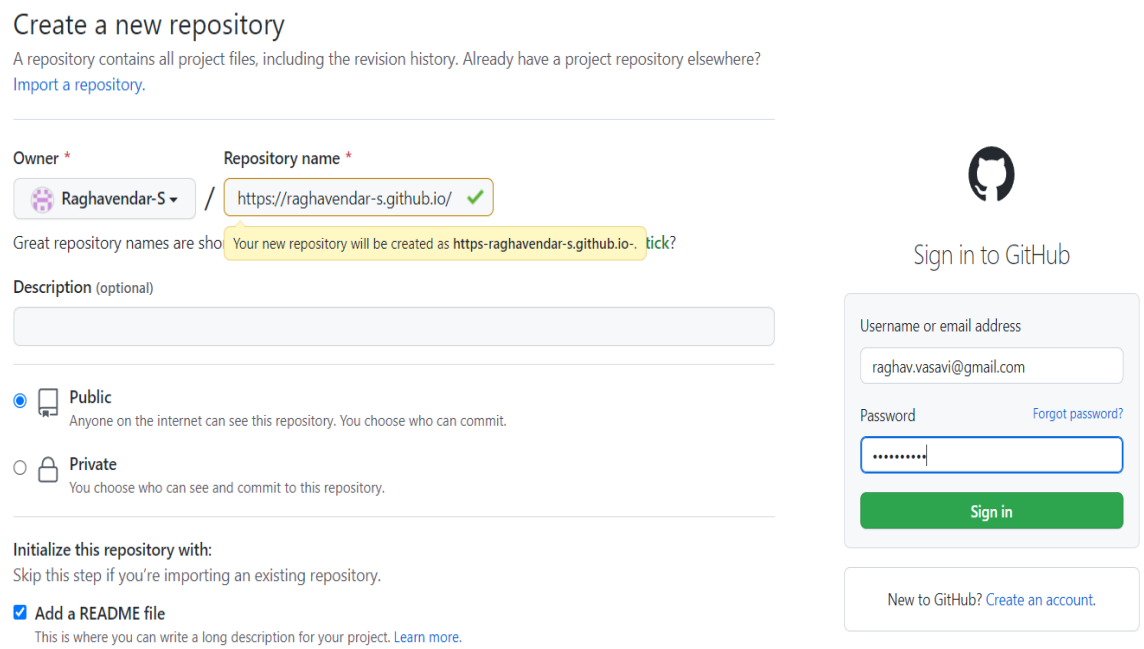
Step 1: Create a website using Html and CSS.

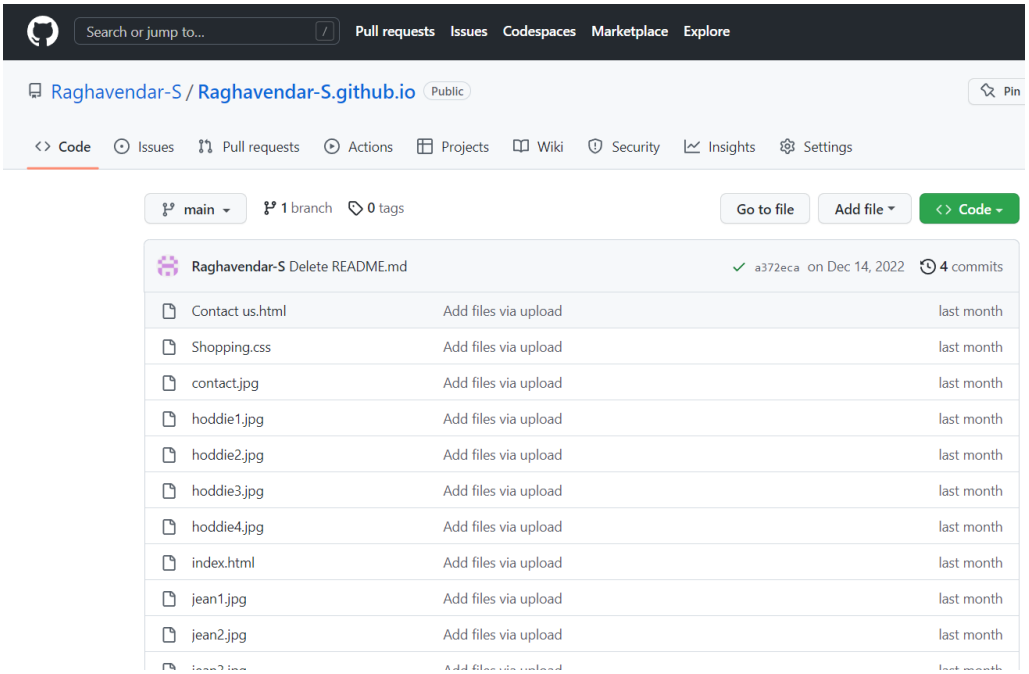
Step 2: Login in to the GitHub and create a new public repository named username.github.io, where username is your username (or organization name).

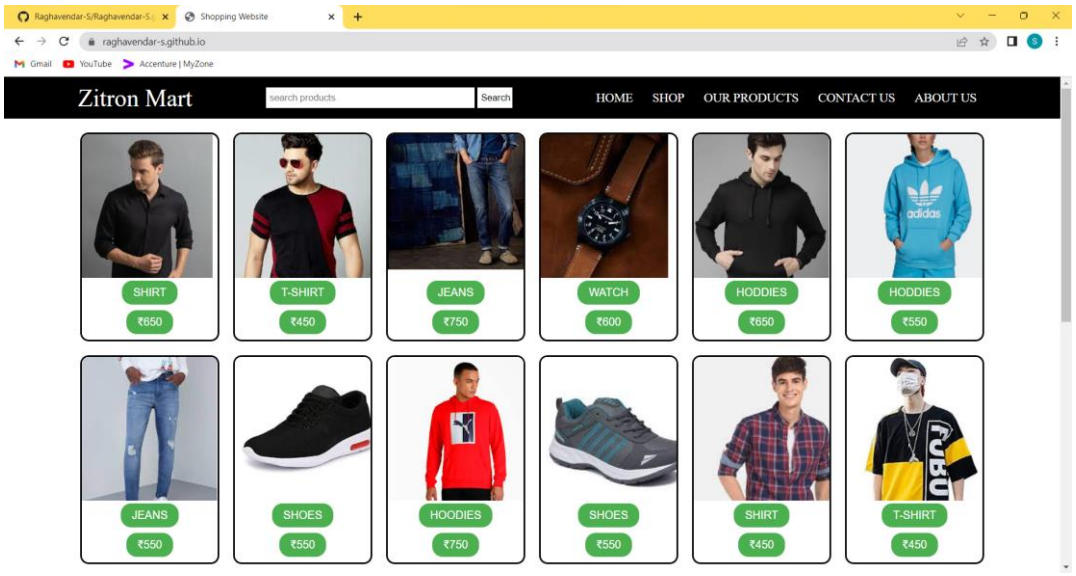
Step 3: After that upload your webpage files to the repository (Note the first page should be named as index.html).

Step 4: Then go to the browser and type the URL name as https://username.github.io/ and the webpage will be displayed.

Output:







Result:

Thus, a webpage is created and deployed it in the cloud.

EX.NO: 2	CREATION OF VIRTUAL MACHINE AND CHECK WHETHER IT HOLDS THE DATA EVEN AFTER RELEASE OF VIRTUAL MACHINE
DATE:	

Aim:

To create a Virtual machine and attach a Virtual Block to check whether it holds the data even after the release of virtual machine.

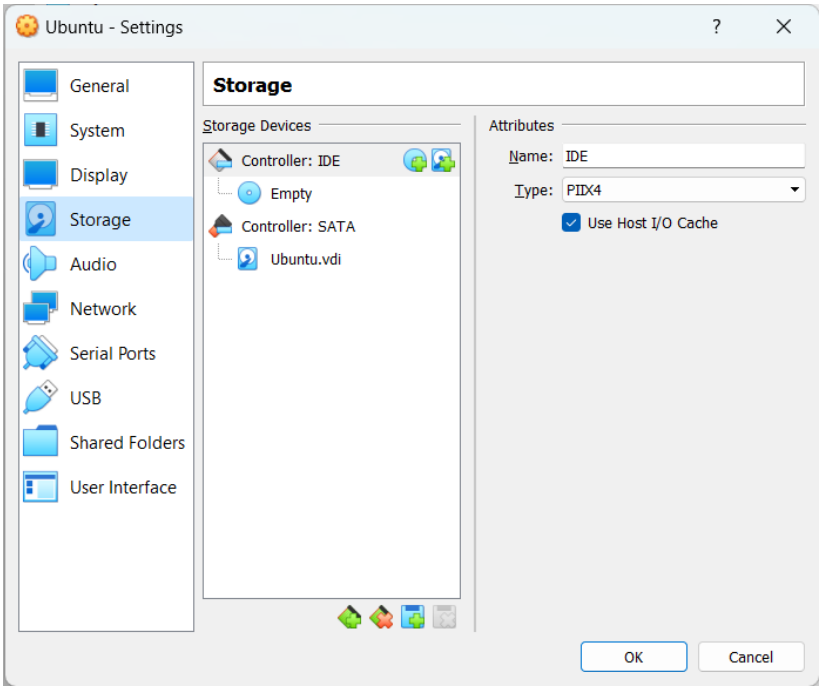
Requirements:

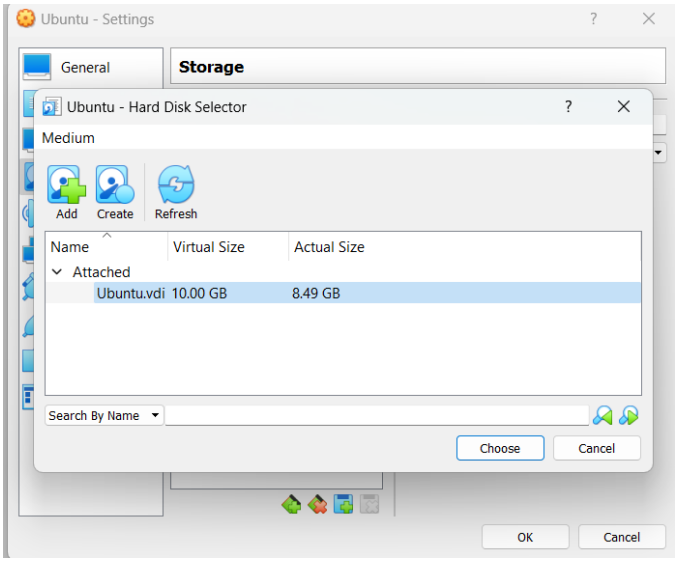
- Oracle virtual box
- Ubuntu ISO file

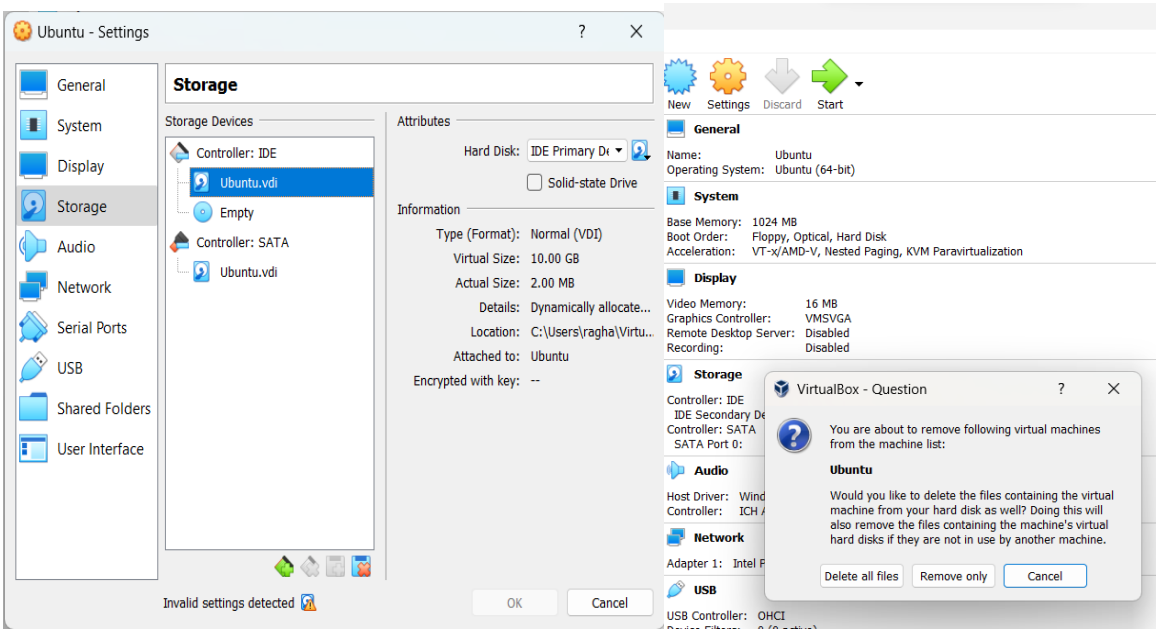
Procedure:

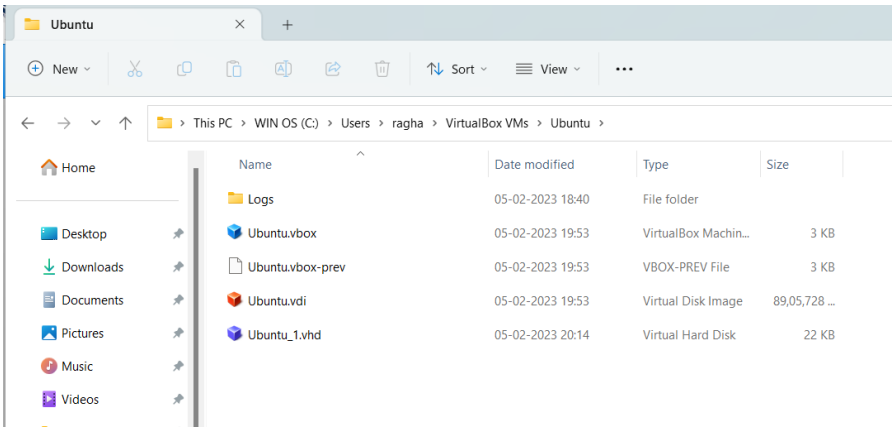
Step 1: Open the virtual box and install the Ubuntu OS in it.
Step 2: Power off the VM which you want to add virtual box
Step 3: Then right click on that VM, select settings.
Step 4: Then click on storage, find controller IDE.
Step 5: In the top right find add hard disk icon, the pop-up window display
Step 6: On that window select create new disk, and then click next and next then finish.
Step 7: Then find attributes icon, hard disk as IDE secondary slave.

Output:









Result:

Thus, a Virtual Block is attached to a Virtual Machine and checked whether it holds the data even after the release of virtual machine.

EX.NO: 3	INSTALLATION OF COMPILER IN A VIRTUAL MACHINE
DATE:	

Aim:

To install a compiler in the virtual Machine and execute a sample program in it.

Requirements:

- VMware Workstation Pro
- Ubuntu ISO file

Procedure:

Step 1: Login to the ubuntu OS and open the terminal.

Step 2: To check whether the gcc compiler is already installed or not use the command:

gcc –version

This command shows the version if gcc is installed or shows error if it is not installed.

Step 3: To install the gcc compiler use the command:

sudo apt-get install gcc

sudo apt-get install build-essential

Step 4: To create a C program file then create a text file with extension .c using the command

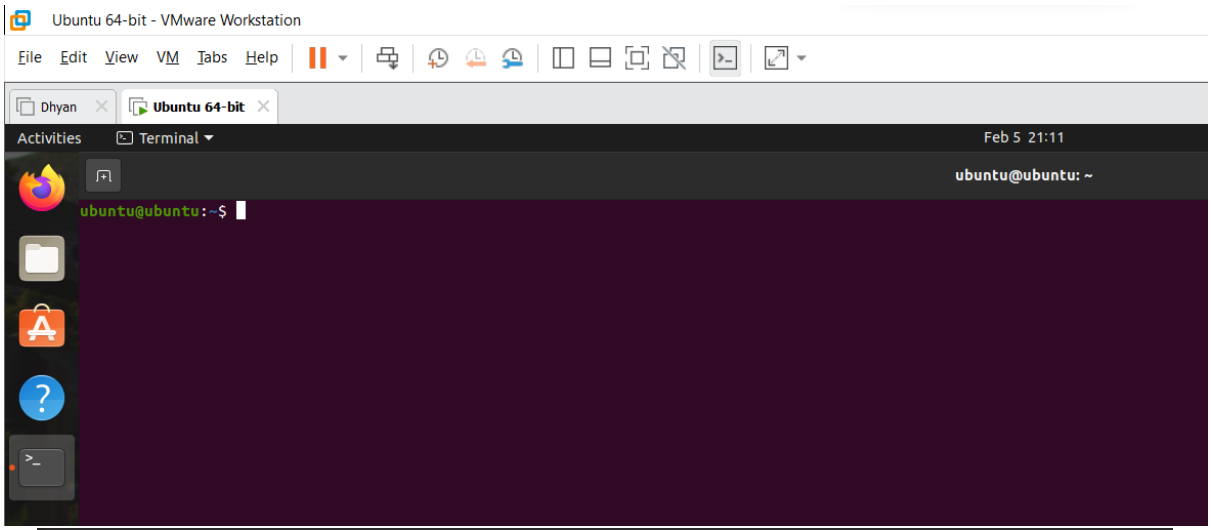
gedit hello.c

Step 5: After creating the c program compile and run the program using the command:

gcc hello.c

./a.out

Output:




```
ubuntu@ubuntu:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
ubuntu@ubuntu:~$ sudo apt-get install gcc
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:9.3.0-1ubuntu2).
gcc set to manually installed.
The following packages were automatically installed and are no longer required:
  linux-headers-5.13.0-30-generic linux-hwe-5.13-headers-5.13.0-30 linux-image-5.13.0-30-generic linux-modules-5.13.0-30-generic linux-modules-extra-5.13.0-30-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 92 not upgraded.
```

```
ubuntu@ubuntu:~$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
The following packages were automatically installed and are no longer required:
  linux-headers-5.13.0-30-generic linux-hwe-5.13-headers-5.13.0-30 linux-image-5.13.0-30-generic linux-modules-5.13.0-30-generic linux-modules-extra-5.13.0-30-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 92 not upgraded.
```

Open hello.c

```c
#include<stdio.h>
int main()
{
        printf("Hello world");
        return 0;
}
```

```
ubuntu@ubuntu:~$ gedit hello.c
^C
ubuntu@ubuntu:~$ gcc hello.c
ubuntu@ubuntu:~$ ./a.out
Hello worldubuntu@ubuntu:~$
```

Result:

Thus, the C Compiler in the Virtual Machine is installed and a sample program is executed successfully.

EX.NO: 4	INSTALLATION OF DOCKER ENGINE AND COMPOSE
DATE:	

Aim:

To install a Docker Engine and compose it.

Requirements:

- Docker Desktop and Compose
- Docker Hub

Description:

Docker:

- Docker is a software platform that allows you to build, test, and deploy applications quickly.
- Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Docker compose:

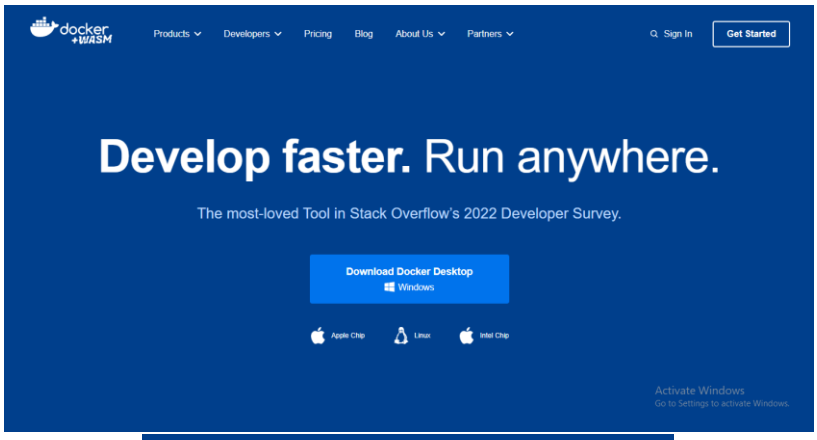
- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a YAML file to configure your application's services.
- Then, with a single command, you create and start all the services from your configuration.
- Compose works in all environments: production, staging, development, testing, as well as CI workflows.

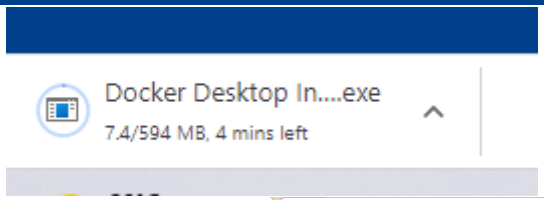
Procedure:

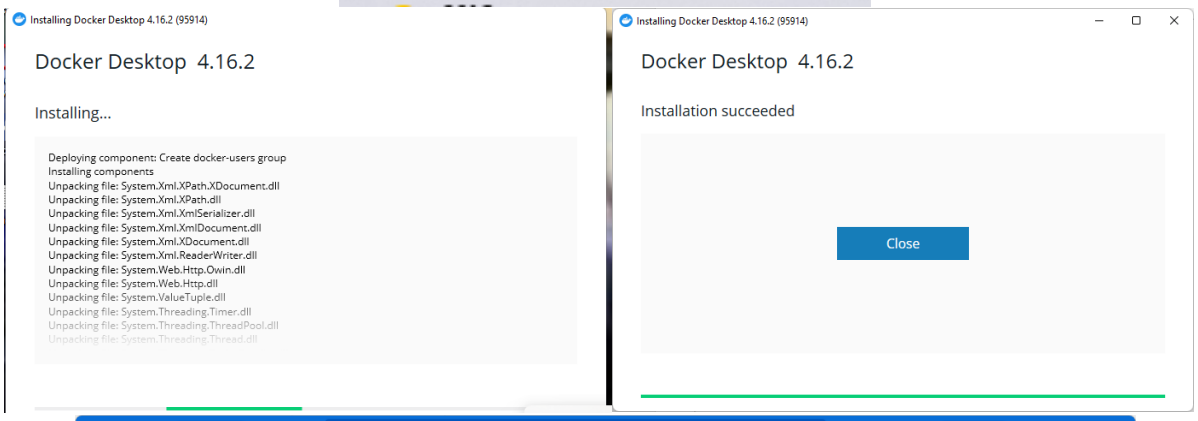
Docker Desktop Installation:

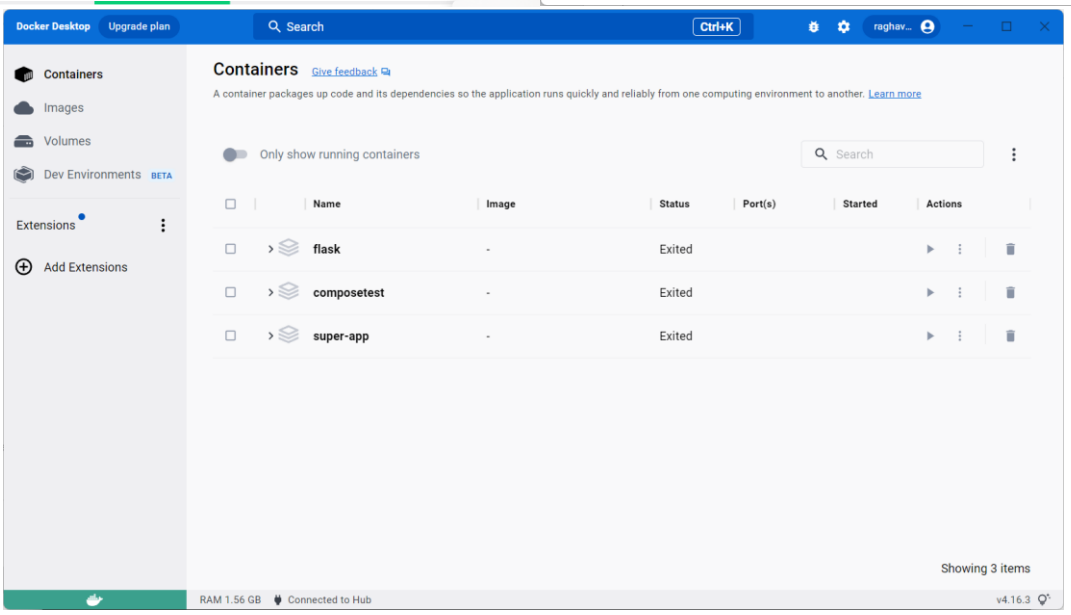
Step 1: Go to the website https://www.docker.com/products/docker-desktop/ and download the docker desktop application for your operating system.
Step 2: Then click on the docker desktop installer and start installing it.
Step 3: After installing, click on Finish Button.
Step 4: Then go to the docker desktop application and accept the terms and conditions and finally your docker desktop is ready to use.

Output:









Result:

Thus, the docker desktop has been installed successfully.

EX.NO: 5	WRITING DOCKER FILE FOR SIMPLE APPLICATION DEVELOPMENT
DATE:	

Aim:

To write a Docker file to develop a simple application.

Requirements:

- Docker Desktop and Compose
- Docker Hub

Procedure:

Step 1: Define the application dependencies

1. Create a directory for the project:

```
$ mkdir composetest
$ cd composetest
```

2. Create a file called app.py in your project directory and paste the following code in:

```
import time
import redis
from flask import Flask
app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)
def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

In this example, redis is the hostname of the redis container on the application's network. We use the default port for Redis, 6379.

3. Create another file called requirements.txt in your project directory and paste the following code in:

```
flask
```

- redis

Step 2: Create a Dockerfile

- The Dockerfile is used to build a Docker image. The image contains all the dependencies the Python application requires, including Python itself.
- In your project directory, create a file named Dockerfile and paste the following code in:

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

This tells Docker to:

- Build an image starting with the Python 3.7 image.
- Set the working directory to /code.
- Set environment variables used by the flask command.
- Install gcc and other dependencies
- Copy requirements.txt and install the Python dependencies.
- Add metadata to the image to describe that the container is listening on port 5000
- Copy the current directory. in the project to the workdir. in the image.
- Set the default command for the container to flask run.

Step 3: Define services in a Compose file

- Create a file called docker-compose.yml in your project directory and paste the following:

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

- This Compose file defines two services: web and redis.
- The web service uses an image that's built from the Dockerfile in the current directory. It then binds the container and the host machine to the exposed port, 8000. This example service uses the default port for the Flask web server, 5000.
- The redis service uses a public Redis image pulled from the Docker Hub registry.

Step 4: Build and run your app with Compose

- From your project directory, start up your application by running docker compose up.

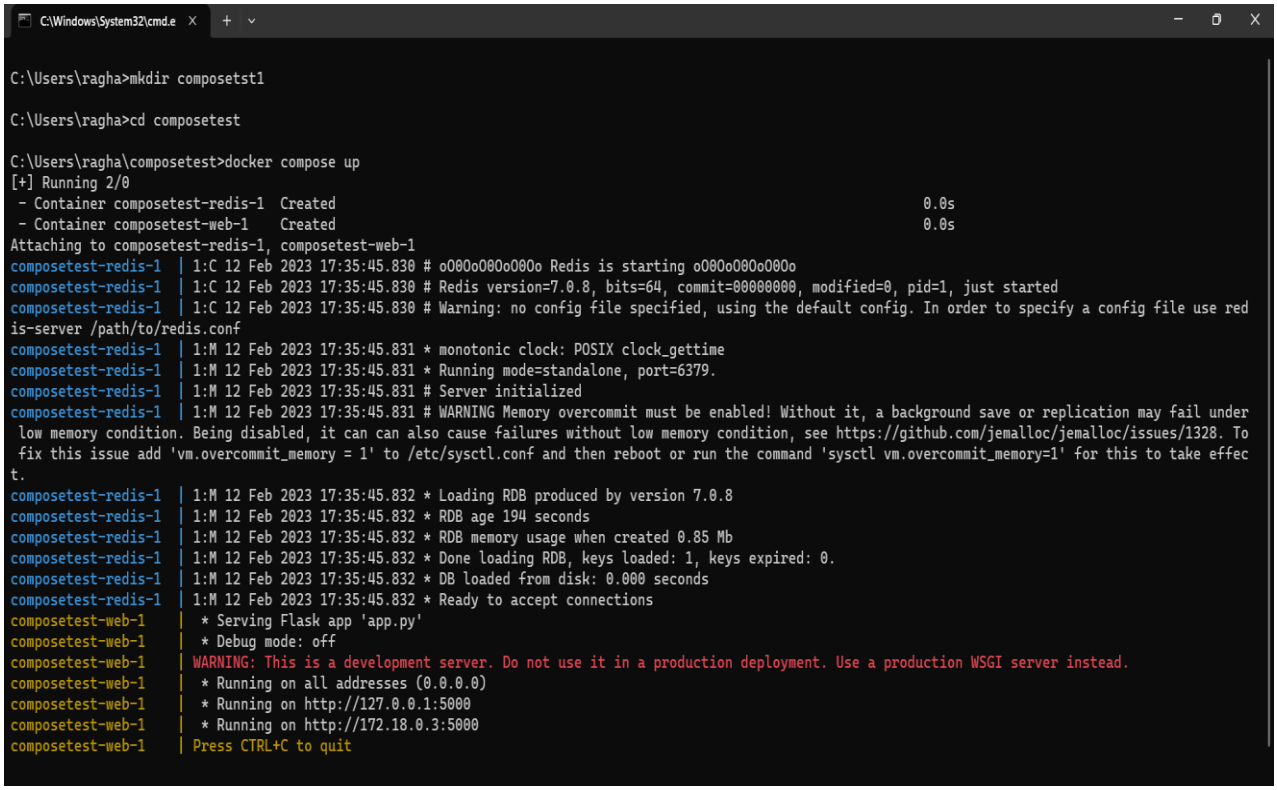
$ docker compose up

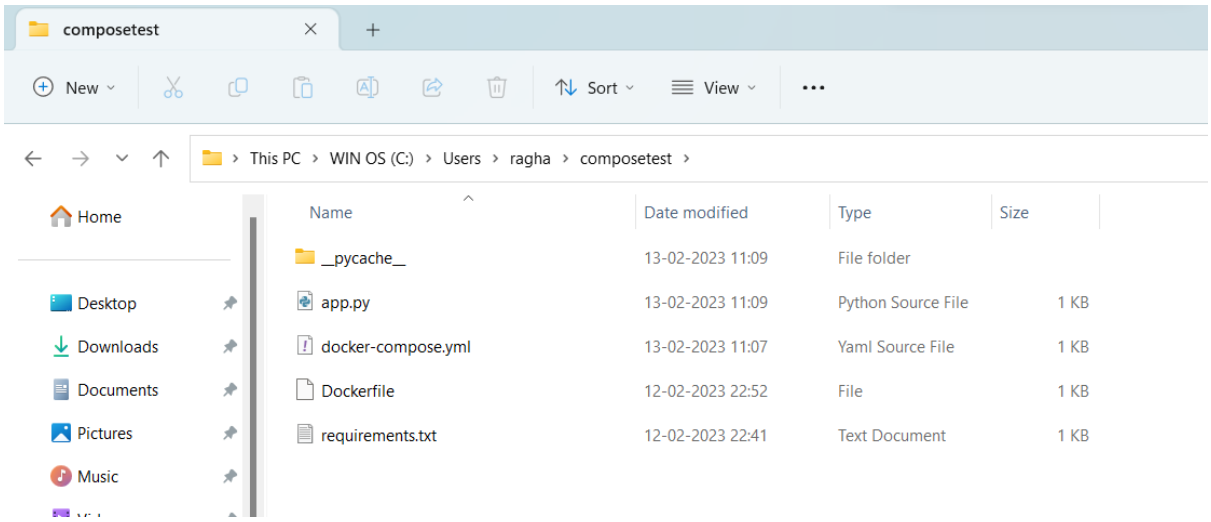
- Compose pulls a Redis image, builds an image for your code, and starts the services you defined. In this case, the code is statically copied into the image at build time.
- Enter http://localhost:8000/ in a browser to see the application running.
- If this doesn't resolve, you can also try http://127.0.0.1:8000.
- You should see a message in your browser saying:

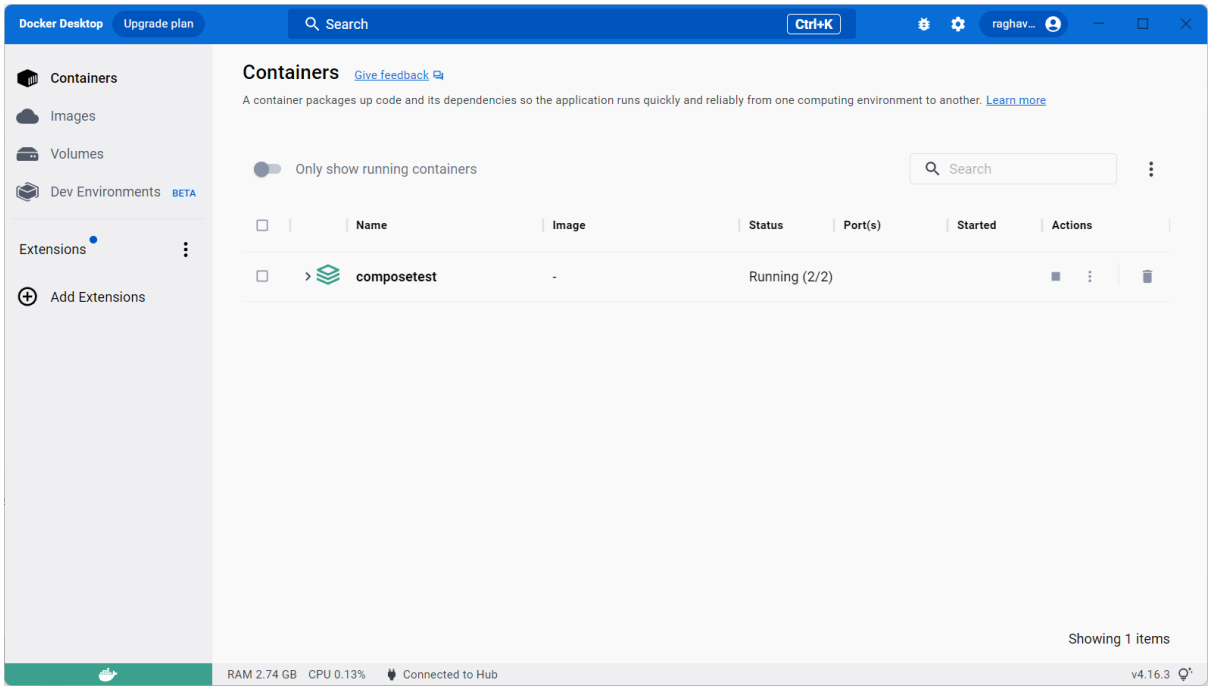
  Hello World! I have been seen 1 time.
- Refresh the page. The number should increment.

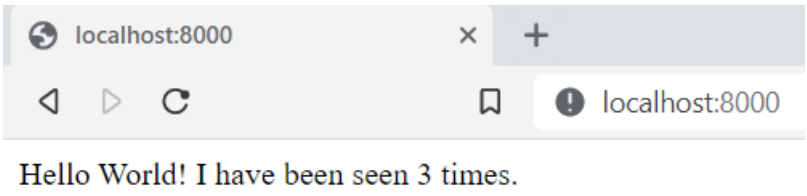
  Hello World! I have been seen 2 times.

Output:









Result:

Thus, a simple application is developed with the help of docker file.

EX.NO: 6	PUSH AND PULL FROM/TO DOCKER HUB
DATE:	

Aim:

To perform a pull and push commands to Docker Hub.

Requirements:

- Docker Desktop and Compose
- Docker Hub

Procedure:

Step 1: First login to your Docker Hub and create a repository named as docker-demo.
Step 2: After that a new repository is created in your docker hub.
Step 3: Create a DockerFile and necessary packages. Start build the docker image using the command

docker build -t your_directory_name:tag_name .

Step 4: Then type the command docker images to check whether the image is successfully created or not.
Step 5: Then create a tag using the following command

docker tag image_id docker_hub_username/repository_name:tag_name

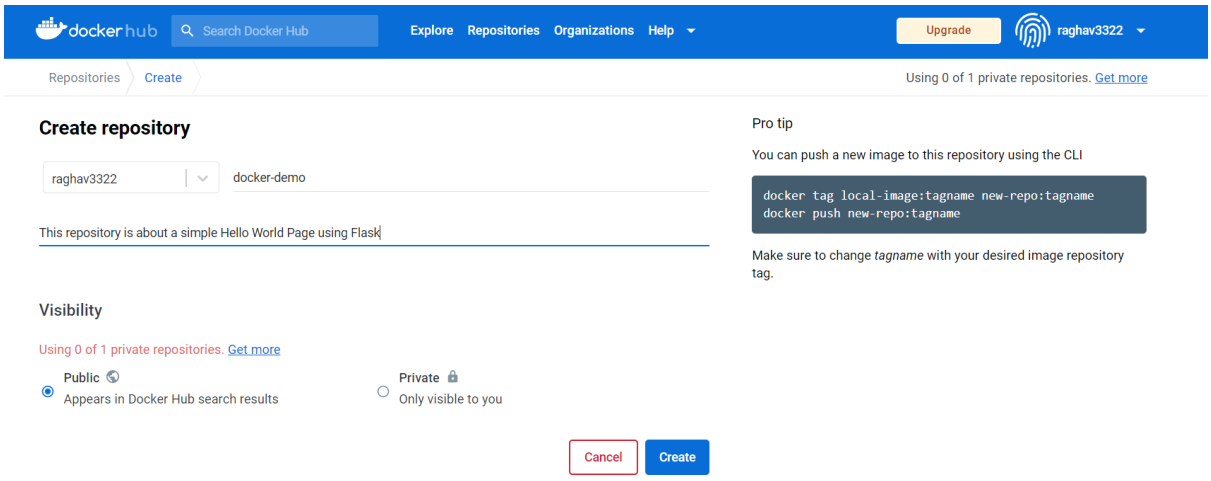
Step 6: Then type the command docker images to check whether the image is successfully created or not.
Step 7: Then push the image, use the following command

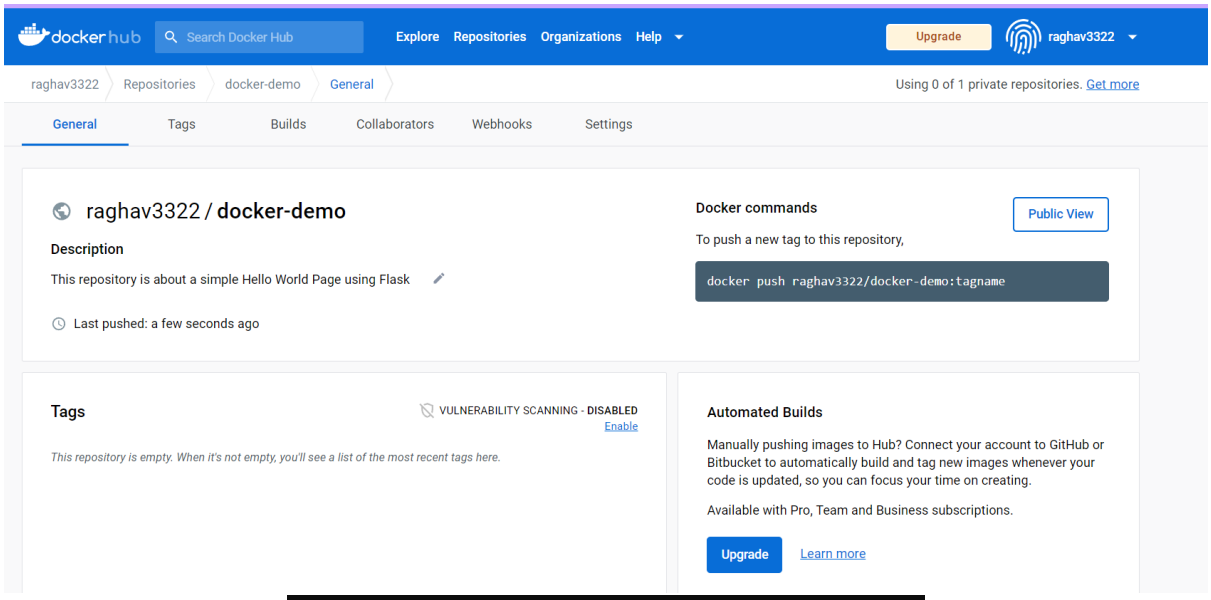
docker push docker_hub_username/repository_name

Step 8: To pull the image from repository, use the following command

docker pull docker_hub_username/repository_name:tag_name

Output:



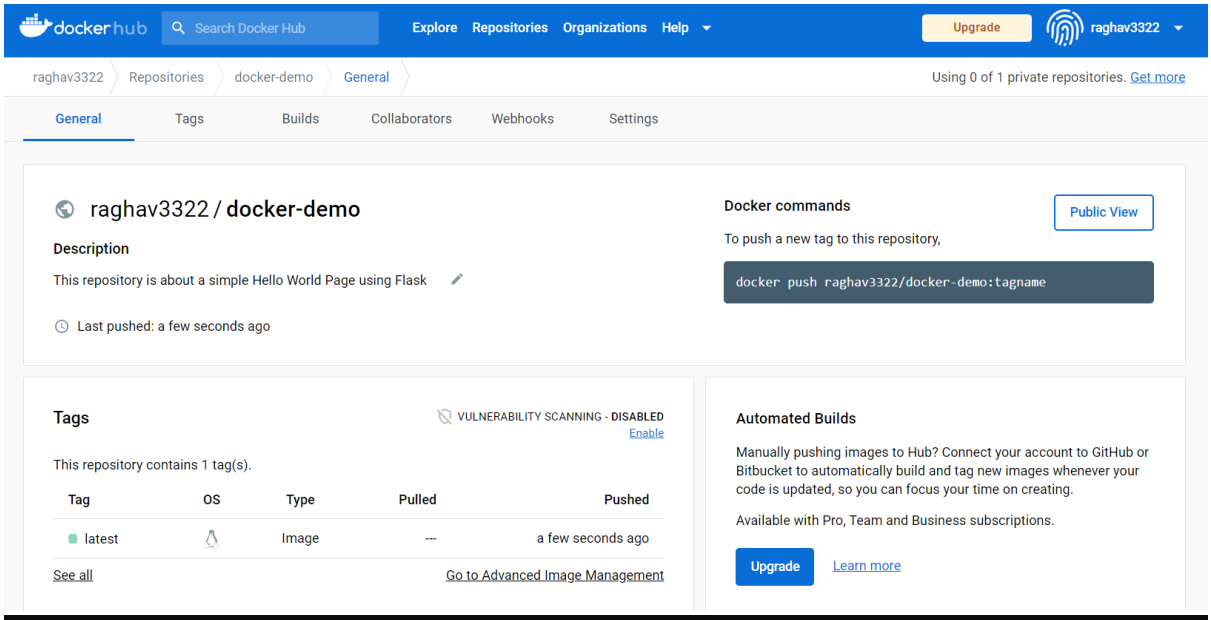



```
C:\Users\ragha\composetest>docker login
Authenticating with existing credentials...
Login Succeeded
```

```
C:\Users\ragha\composetest>docker build -t composetest:latest .
[+] Building 3.4s (17/17) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 32B                                                                                 0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                          1.9s
 => [auth] docker/dockerfile:pull token for registry-1.docker.io                                                    0.0s
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5feceb7372a0817649ecb2724562a38360f4  0.0s
 => [internal] load build definition from Dockerfile                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.7-alpine                                                1.3s
 => [auth] library/python:pull token for registry-1.docker.io                                                       0.0s
 => [1/6] FROM docker.io/library/python:3.7-alpine@sha256:962a81cc5763f49326a48b5d36be1c3e824a53c05d9f57f054dcd8d   0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 212B                                                                                   0.0s
 => CACHED [2/6] WORKDIR /code                                                                                      0.0s
 => CACHED [3/6] RUN apk add --no-cache gcc musl-dev linux-headers                                                  0.0s
 => CACHED [4/6] COPY requirements.txt requirements.txt                                                             0.0s
 => CACHED [5/6] RUN pip install -r requirements.txt                                                                0.0s
 => CACHED [6/6] COPY . .                                                                                           0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:392257fb07a1888b90f81886b6fe8960530b1213a89c6b8987c14dfe55d92884                        0.0s
 => => naming to docker.io/library/composetest:latest                                                               0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\ragha\composetest>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
composetest                 latest    392257fb07a1   38 minutes ago   215MB
```

```
C:\Users\ragha\composetest>docker tag 392257fb07a1 raghav3322/docker-demo:latest

C:\Users\ragha\composetest>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
composetest                 latest    392257fb07a1   39 minutes ago   215MB
raghav3322/docker-demo      latest    392257fb07a1   39 minutes ago   215MB
```

```
C:\Users\ragha\composetest>docker push raghav3322/docker-demo
Using default tag: latest
The push refers to repository [docker.io/raghav3322/docker-demo]
c5bf3ddeac9a: Layer already exists
fc124490f06e: Layer already exists
9e689f319f4a: Layer already exists
9610745e2833: Pushed
1d06a5ea0a00: Layer already exists
157fa08f39ef: Layer already exists
0eb3a89ef0fe: Layer already exists
c9e377358a79: Layer already exists
f7cd9720fc7e: Layer already exists
7cd52847ad77: Layer already exists
latest: digest: sha256:710aac2ed70229be8da28e91a3b6d575989ce2f42ccd0efaaf6331b6bf088e4d size: 2413
```




```
C:\Users\ragha>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
composetest                 latest    392257fb07a1   42 minutes ago   215MB
raghav3322/docker-demo      latest    392257fb07a1   42 minutes ago   215MB
flask-web                   latest    52bdd8b82ca7   4 days ago       60.4MB
composetest-web             latest    e263079e4ffa   4 days ago       215MB
super-app-super-app-php     latest    3bece0924f89   4 days ago       455MB
super-app-super-app-dotnet  latest    e412b7ee8401   5 days ago       217MB
super-app-super-app-python  latest    6a7d3411b7af   5 days ago       436MB
super-app-super-app-node    latest    43afee1c1ccf   5 days ago       248MB
<none>                      <none>    9482071cf9d5   8 days ago       215MB
redis                       alpine    aeeb92ae6202   9 days ago       29.9MB
mysql                       8.0.28    f2ad9f23df82   10 months ago    521MB

C:\Users\ragha>docker rmi raghav3322/docker-demo
Untagged: raghav3322/docker-demo:latest
Untagged: raghav3322/docker-demo@sha256:710aac2ed70229be8da28e91a3b6d575989ce2f42ccd0efaaf6331b6bf088e4d
```

```
C:\Users\ragha>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
composetest                 latest    392257fb07a1   43 minutes ago   215MB
flask-web                   latest    52bdd8b82ca7   4 days ago       60.4MB
composetest-web             latest    e263079e4ffa   4 days ago       215MB
super-app-super-app-php     latest    3bece0924f89   4 days ago       455MB
super-app-super-app-dotnet  latest    e412b7ee8401   5 days ago       217MB
super-app-super-app-python  latest    6a7d3411b7af   5 days ago       436MB
super-app-super-app-node    latest    43afee1c1ccf   5 days ago       248MB
<none>                      <none>    9482071cf9d5   8 days ago       215MB
redis                       alpine    aeeb92ae6202   9 days ago       29.9MB
mysql                       8.0.28    f2ad9f23df82   10 months ago    521MB

C:\Users\ragha>docker pull raghav3322/docker-demo:latest
latest: Pulling from raghav3322/docker-demo
Digest: sha256:710aac2ed70229be8da28e91a3b6d575989ce2f42ccd0efaaf6331b6bf088e4d
Status: Downloaded newer image for raghav3322/docker-demo:latest
docker.io/raghav3322/docker-demo:latest

C:\Users\ragha>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
composetest                 latest    392257fb07a1   43 minutes ago   215MB
raghav3322/docker-demo      latest    392257fb07a1   43 minutes ago   215MB
```

Result:

Thus, the pull and push commands are performed in docker hub.

EX.NO: 7	RUNNING MULTIPLE DOCKER CONTAINERS USING DOCKER COMPOSE
DATE:	

Aim:

To run multiple Docker containers using docker compose.

Requirements:

- Docker Desktop and Compose
- Docker Hub

Procedure:

Step 1: Create a folder named react-java-mysql. Next, create a compose.yaml file.
Step 2: Create 3 folders frontend, backend and db where frontend consist of react app, backend consists of Spring (Java) and Maria as database.
Step 3: Create a docker-compose.yml file which consist of three services frontend, backend and database.
Step 4: After that compose the container using the command docker compose up. Then go to the localhost:3000 port. You can see your react app.

Program:

Compose.yml

```
services:
  backend:
    build: backend
    restart: always
    secrets:
      - db-password
    environment:
      MYSQL_HOST: db
    networks:
      - react-spring
      - spring-mysql
    depends_on:
      db:
        condition: service_healthy
  db:
    # We use a mariadb image which supports both amd64 & arm64 architecture
    image: mariadb:10.6.4-focal
    # If you really want to use MySQL, uncomment the following line
    #image: mysql:8.0.19
    environment:
      - MYSQL_DATABASE=example
      - MYSQL_ROOT_PASSWORD_FILE=/run/secrets/db-password
    restart: always
```

```
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "127.0.0.1", "--silent"]
      interval: 3s
      retries: 5
      start_period: 30s
    secrets:
      - db-password
    volumes:
      - db-data:/var/lib/mysql
    networks:
      - spring-mysql
  frontend:
    build:
      context: frontend
      target: development
    ports:
      - 3000:3000
    volumes:
      - ./frontend/src:/code/src
      - /project/node_modules
    networks:
      - react-spring
    depends_on:
      - backend
    expose:
      - 3306
      - 33060
volumes:
  db-data: {}
secrets:
  db-password:
    file: db/password.txt
networks:
  react-spring: {}
  spring-mysql: {}
```

App.tsx

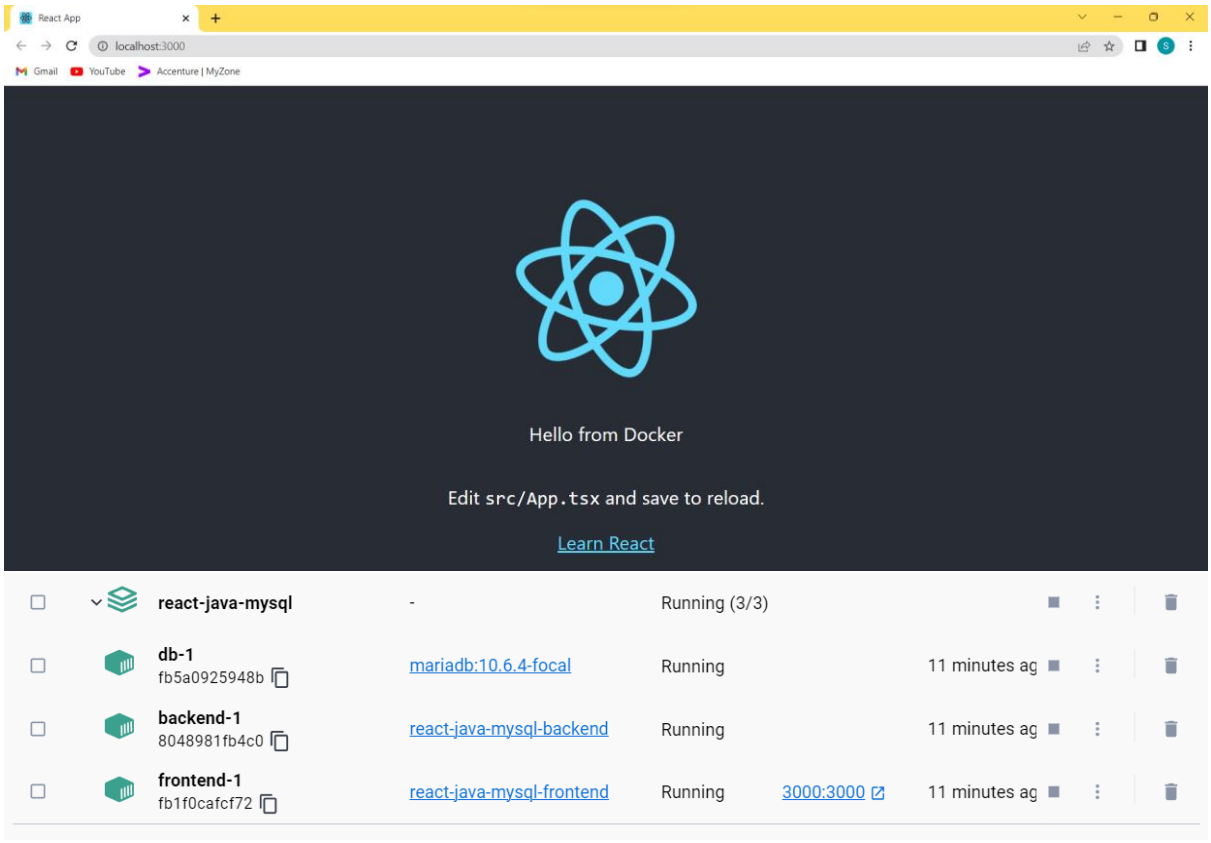
```
import React, { useEffect, useState } from "react";
import logo from "./logo.svg";
import "./App.css";
type Greeting = {
  id: number;
  name: string;
};
function App() {
  const [greeting, setGreeting] = useState<Greeting>();
  useEffect(() => {
```

```
    fetch("/api")
      .then(res => res.json())
      .then(setGreeting)
      .catch(console.error);
  }, [setGreeting]);
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        {greeting ? (
          <p>Hello from {greeting.name}</p>
        ) : (
          <p>Loading...</p>
        )}
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}
export default App;
```

Output:

```
C:\Users\ragha\react-java-mysql>docker compose up
[+] Running 11/11
 - db Pulled                                                                                                    42.8s
   - 7b1a6ab2e44d Pull complete                                                                                 18.9s
   - 034655750c88 Pull complete                                                                                 19.0s
   - f0b757a2a0f0 Pull complete                                                                                 19.2s
   - 5c37daf8b6b5 Pull complete                                                                                 19.3s
   - b4cd9409b0f6 Pull complete                                                                                 19.3s
   - dbcda06785eb Pull complete                                                                                 19.6s
   - a34cd90f184c Pull complete                                                                                 19.6s
   - fd6cef4ce489 Pull complete                                                                                 19.7s
   - 3cb89a1550ea Pull complete                                                                                 38.7s
   - df9f153bd930 Pull complete                                                                                 38.7s
[+] Building 337.9s (39/39) FINISHED
 => [react-java-mysql-backend internal] load build definition from Dockerfile                                    0.0s
 => => transferring dockerfile: 1.08kB                                                                           0.0s
 => [react-java-mysql-frontend internal] load build definition from Dockerfile                                   0.0s
 => => transferring dockerfile: 709B                                                                             0.0s
 => [react-java-mysql-backend internal] load .dockerignore                                                       0.0s
 => => transferring context: 2B                                                                                  0.0s
```

Name	Date modified	Type	Size
.docker	21-02-2023 19:24	File folder	
backend	21-02-2023 19:24	File folder	
db	21-02-2023 19:24	File folder	
frontend	21-02-2023 19:24	File folder	
compose.yaml	10-01-2023 03:00	Yaml Source File	2 KB
output.jpg	10-01-2023 03:00	JPG File	36 KB
README.md	10-01-2023 03:00	Markdown Source ...	4 KB



Result:

Thus, multiple containers have been successfully created using docker compose.

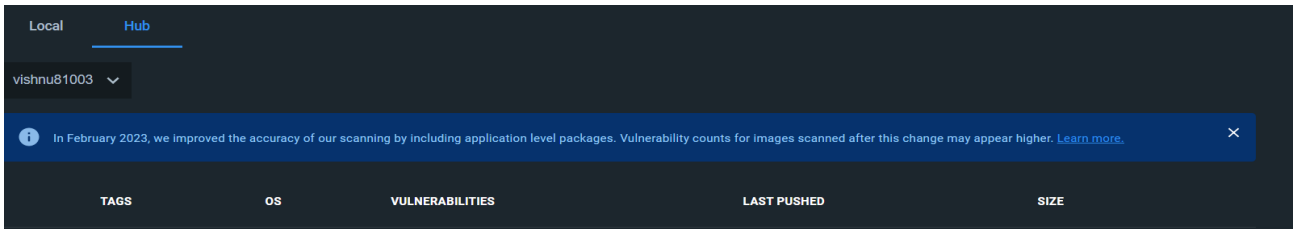
EX.NO: 8	EXPERIMENTS ON DOCKER SWARM
DATE:	

Aim:

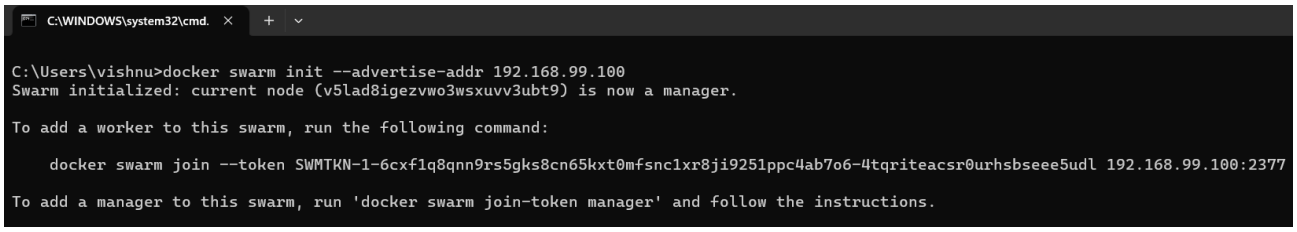
To create and manage docker swarm using Docker.

Procedure:

Step 1: Start.
Step 2: Install Docker Desktop and Log in to your account on it.

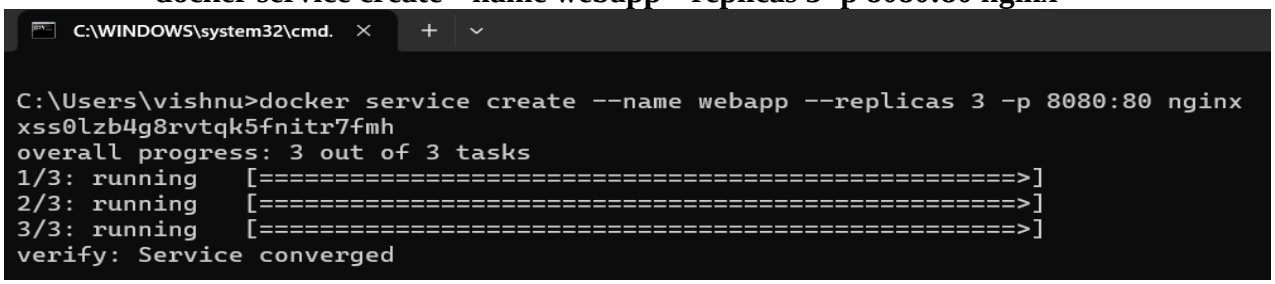


Step 3: To create a swarm specify an IP Address in which the swarm is needed to be created (Be careful that creating a swarm required firewall permissions).

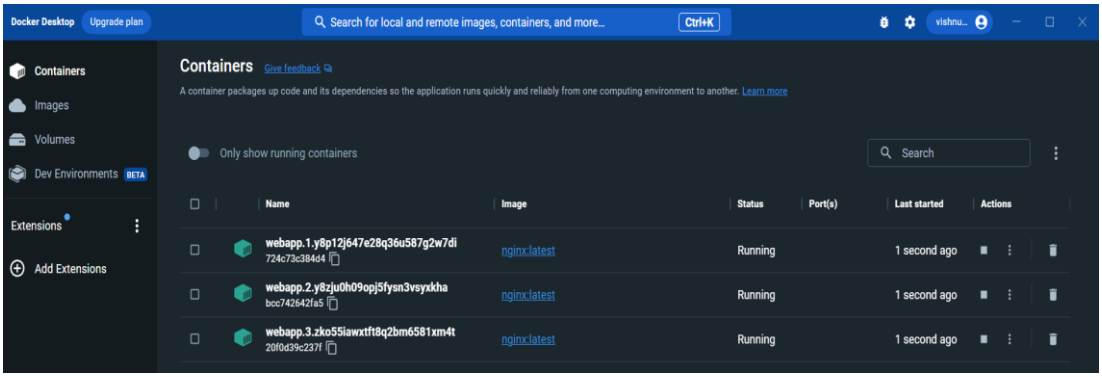


Step 4: Let's create a swarm service with the nginx image(ver.1.17.6) in the port 8080 named as webapp

docker service create --name webapp --replicas 3 -p 8080:80 nginx



Docker Reflection:



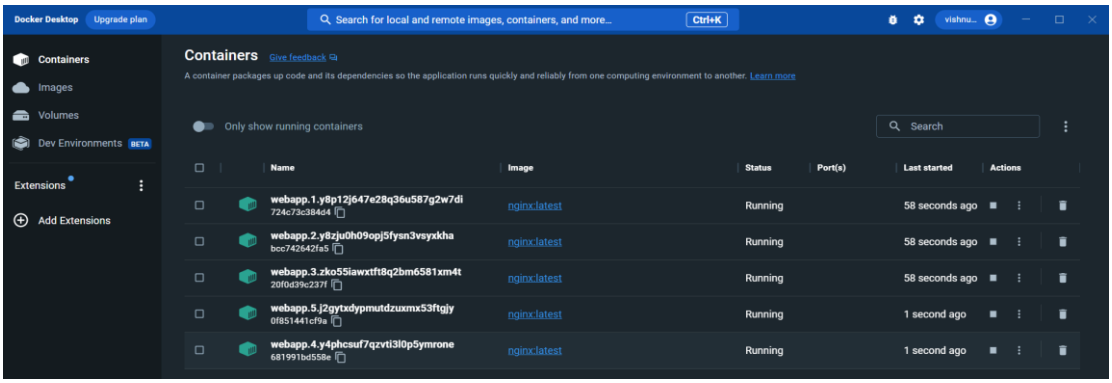
Step 5: The process we perform is that we create a copy of the images in the swarm using scaling command.

docker service scale webapp=5

```
C:\Users\vishnu>docker service scale webapp=5
webapp scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
```

Two new containers will be formed as a replica of the previous nginx images

Docker reflection:



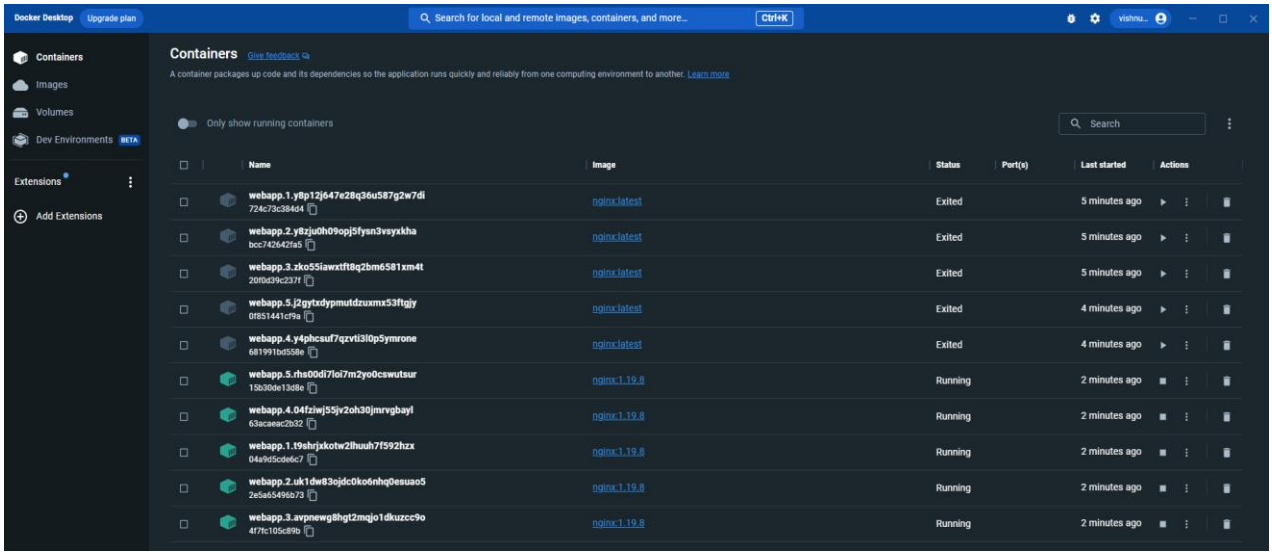
Step 6: Update the swarm containers with the new containers with the nginx image's next version 1.19.8

docker service update --image nginx:1.19.8 webapp

```
C:\Users\vishnu>docker service update --image nginx:1.19.8 webapp
webapp
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
```

This process would stop all the running containers and run new containers with the updated versions on the swarm

Docker Reflection:



Step 7: We can manually delete all the containers in the docker desktop itself. but in the instance of swarm, we can't delete it directly so, we need find the swarm in which those images are created. (If u try to delete the containers in docker desktop created using the swarm it will automatically replicate itself. you could never stop the container from running)

Docker services ls –filter name=webapp

```
C:\Users\vishnu>docker service ls --filter name=webapp
ID             NAME      MODE          REPLICAS   IMAGE          PORTS
xss0lzb4g8rv   webapp    replicated    5/5        nginx:1.19.8   *:8080->80/tcp
```

This displays all the details of the swarm

Step 8: Find all the containers running on the Swarm

Docker ps -a

```
C:\Users\vishnu>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
543da95ba4d7   nginx:1.19.8   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    webapp.1.oiksgucacdlss508vmdoxt0jb
ca0d19d2ee12   nginx:1.19.8   "/docker-entrypoint.…"   3 minutes ago        Up 3 minutes        80/tcp    webapp.3.6xxjrvmankhtl8ngaj81qiqkv
0c9dd5fe004c   nginx:1.19.8   "/docker-entrypoint.…"   4 minutes ago        Up 4 minutes        80/tcp    webapp.2.tvcqthumoevn9wobsdsagouzq
2839b7ddcf01   nginx:1.19.8   "/docker-entrypoint.…"   4 minutes ago        Up 4 minutes        80/tcp    webapp.5.ui5d7gdqm4zv4lx9hsardy470
32625b9ccc9a   nginx:1.19.8   "/docker-entrypoint.…"   4 minutes ago        Up 4 minutes        80/tcp    webapp.4.8jnrxk41264bce9khdbraz16r
```

This command displays all the running and stopped containers in the swarm. "-a" is used to display all the containers that aren't in the working(running) state.

Step 9: After the tasks is been completed exit from the swarm to relieve the IP Address.

Docker swarm leave --force

```
C:\Users\vishnu>docker swarm leave --force
Node left the swarm.
```

If you don't leave the swarm, you couldn't be able to run any other images or process in the local machine/docker.

Step 10: After the process is completed remove all the volumes of the containers so that the spaces occupied by those images and containers would be freed up.

Docker system prune -a --volumes

```
C:\Users\vishnu>docker system prune -a --volumes
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all volumes not used by at least one container
  - all images without at least one container associated to them
  - all build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
f0b41de2721db394d9f6d641876c84444afebf7c42910dbf5a33c707bb9330e1
ec6e0ad910185a3defb9cbe26406656f1fb38088652490973ab637f49fbba6b489
15a89e1945c4a34852e86327e74d0d7a69b47519bebc9c48b909a7df95167a9e
6c8148bba1d429ad21b08385d19af4863c7eaf31718b6d617488f824eadc79f1
2f8189f55dce587884ce3eb824e1a5b6f3a2b69785be8e9f1f02c2b2f7429f8a
52e51921d4ceea2f15e374ce35a9c737b9f2058b538875f857e410a6ace3b850
bd7dfe28a225b8960799c9d52c05f3d5b727f535c89db3845032ba681005a442
ac467f75457fe322b4780a2d7d337a19805e836787cf25430ac039f5030d9c91
9365526cc832fafce7d61f485973a856e4c02e9875769b6d2d0de632d8aba5714
c066d3efc3e3190420ca68feb383d65fb432526e60ca288eb59ee6bda12fcb9e
1165190a1ab3908bfcdd5f2d5277ea94d74e3aeae18d7e559d5c25094 9db763
3c51406ef6c9c3634f4163bce7d7402db6504b46fe84451ee57ed88a2999d58a
b4e67601a23074dfe7b2f8b19674a87a505f515fee6956c8e974cce5d3cfb5b
93f4fd4ae13a45f053a9b16853310de4aa282fb7b4a29c7d39bda0d59a591e5e
f905ebb4435e5ceb9c7cf5e19c675216ae61cd81499cf9d4da0de2ef6e12a701
7a0b0b57ec9513a420d34ca977ec053a61a80a6ff6d74b201d5ad6cccdf646fc
6028e777b61f5528331b5f7e22abe3b4910cbdea6649bcb2a64bdf42eeb1c448
27fdd4d56e46915a4dbf64e241cc26d039a9a8991611caa0ab3c8bbc1d3688bc
c423fa36b886fb7b61f0f99153f1edabf36c10c3f87c8b6f346bccdf36ca6bd0
dbb87321284fd317694cfb7acbffde8277c15e6d4af865b28c5ffe168a2eb017
34e8d671107491240ecfd433821751 7bc54284ed81a406763e7097db7978a2b5
bd2d823b3c244a9a8be849e81ca3563bc75e99a0517973fd7ca1c6a4b436252b
dd0a0b326ef1edcf8e51e716c1f732b5f9a034def9a114ad59120def1d17a8fa
84bc021063e0ea37e5f1ae43a00f89c9047797e9a8a9e52814054bf5cd1b5a00e
8e3a006092cf166de04dbfe41791d1c499f81958a3165942c7768223dc9bb084

Deleted Networks:
my-network
```

All the disk spaces occupied by the swarms' and its containers are removed and deleted completely.

Result:

Thus, the Docker Swarm has been created and executed successfully.

EX.NO: 9	DEVELOPMENT OF SIMPLE EXPERIMENTS USING MINIKUBE
DATE:	

Aim:

To develop simple experiment using Kubernetes Minikube.

Procedure:

Step 1: Start.

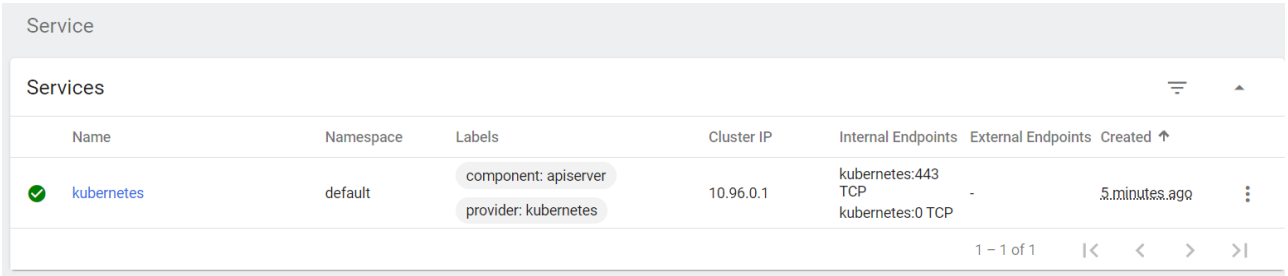
Step 2: Install Kubernetes (Kubectl) from the official documentation page and then install Minikube in the system and set the path for the environment variable.

Step 3: Start the Minikube cluster

Minikube start

```
$ start.sh
Starting Kubernetes...minikube version: v1.18.0
commit: ec61815d60f66a6e4f6353030a40b12362557caa-dirty
* minikube v1.18.0 on Ubuntu 18.04 (kvm/amd64)
* Using the none driver based on existing profile

X The requested memory allocation of 2200MiB does not leave room for system overhead (total system memory: 2460MiB). You may face stability issues.
* Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

* Starting control plane node minikube in cluster minikube
* Running on localhost (CPUs=2, Memory=2460MB, Disk=194868MB) ...
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring local host environment ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
  - Using image k8s.gcr.io/metrics-server-amd64:v0.2.1
* The 'metrics-server' addon is enabled
  - Using image kubernetesui/dashboard:v2.1.0
  - Using image kubernetesui/metrics-scraper:v1.0.4
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


* The 'dashboard' addon is enabled
Kubernetes Started
```

Kubernetes Dashboard:



Step 4: If the cluster is started correctly then, obtain the information of all the clusters running in that deployment

Kubectl cluster-info

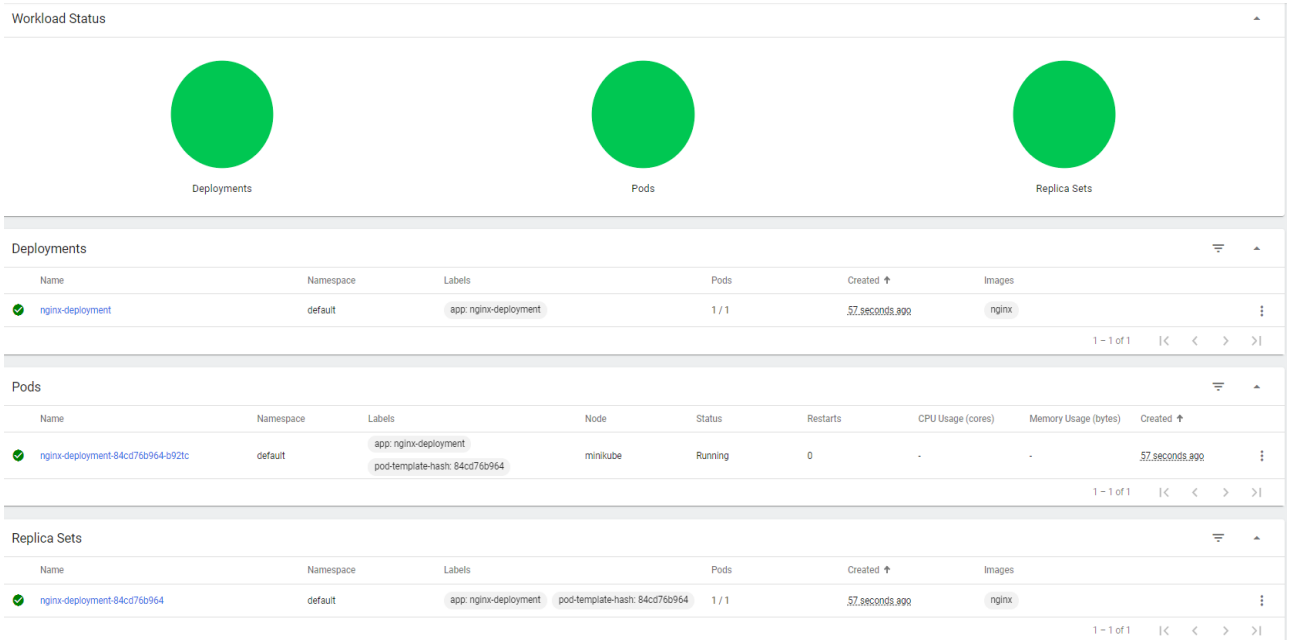
```
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.14:8443
KubeDNS is running at https://10.0.0.14:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

Step 5: Create an image in the deployment. nginx is the deployment we've created here

kubectl create deployment nginx-deployment --image=nginx

```
$ kubectl create deployment nginx-deployment --image=nginx
deployment.apps/nginx-deployment created
```

Kubernetes Dashboard:

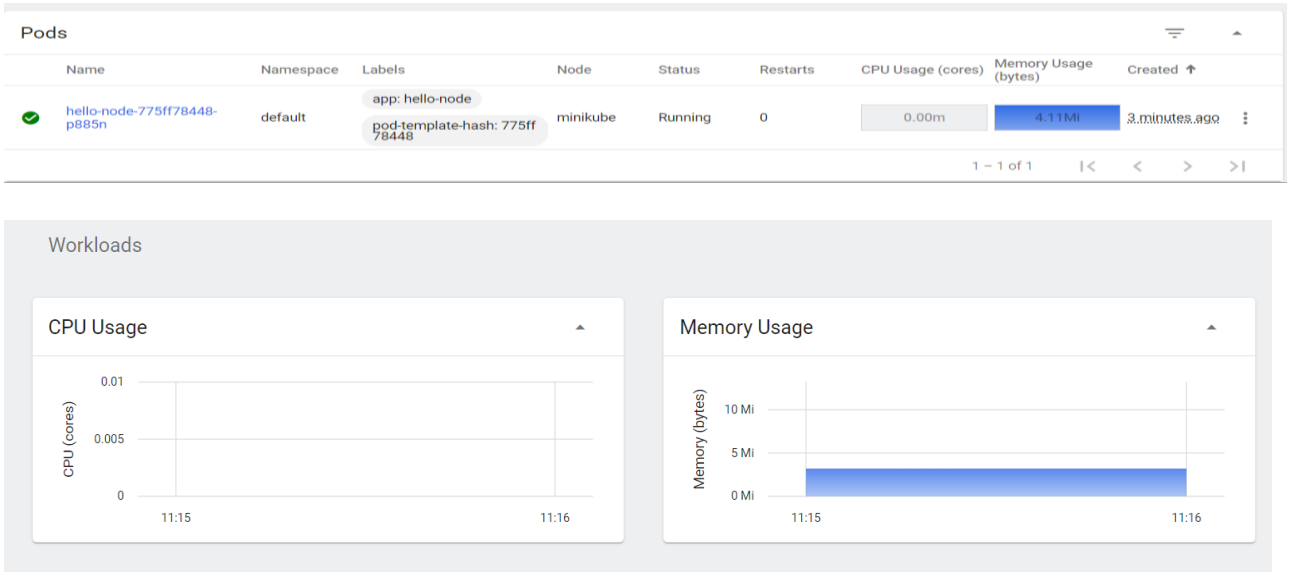


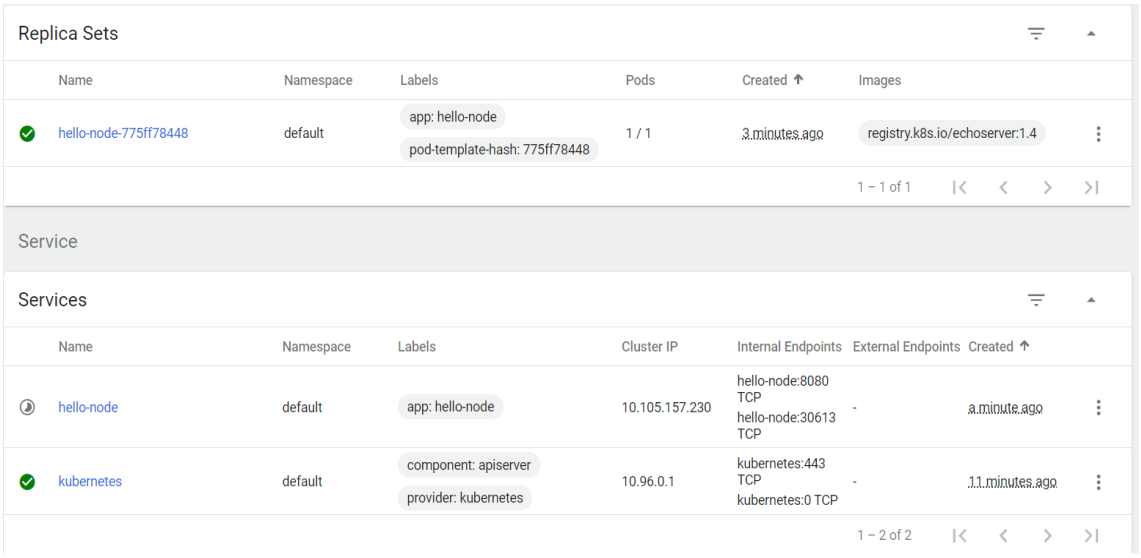
Step 6**:** Obtain all the pods from the deployment. That also displays all the secrets, nodes, Replica sets etc.

kubectl get pods

```
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-6799fc88d8-lvcps              1/1     Running   0          45s
```

Kubernetes Dashboard:





Step 7: Display all the deployments running on the port 80**.**

kubectl expose deployment nginx-deployment --port=80 --type=NodePort

```
|----------------------|------------------------------|--------------|-----------------------|
|      NAMESPACE       |             NAME             | TARGET PORT  |          URL          |
|----------------------|------------------------------|--------------|-----------------------|
| default              | kubernetes                   | No node port |                       |
| kube-system          | kube-dns                     | No node port |                       |
| kube-system          | metrics-server               | No node port |                       |
| kubernetes-dashboard | dashboard-metrics-scraper    | No node port |                       |
| kubernetes-dashboard | kubernetes-dashboard         | No node port |                       |
| kubernetes-dashboard | kubernetes-dashboard-katacoda |           80 | http://10.0.0.8:30000 |
|----------------------|------------------------------|--------------|-----------------------|
```

Result:

Thus, the experiment on Minikube is completed and executed successfully.

DATE:	CONTENT BEYOND SYLLABUS STUDY ON DEVOPS

Aim:

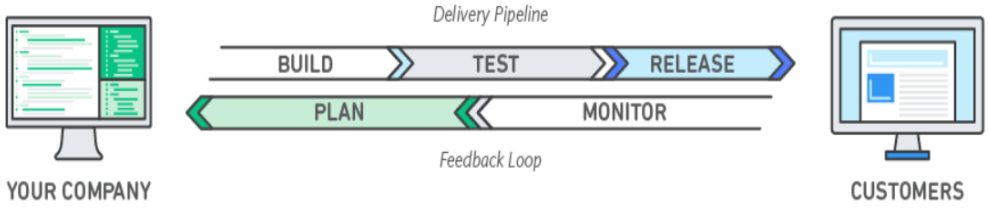
To study about purpose of cloud computing on DevOps.

Description:

DevOps:

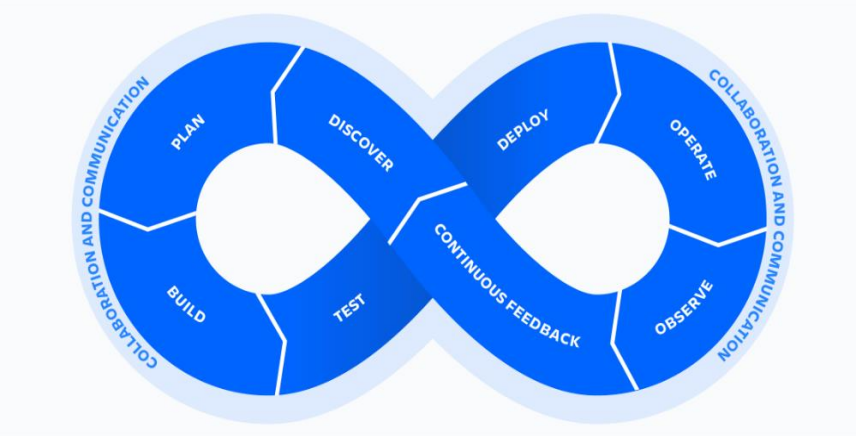
- DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams. It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.
- The DevOps movement began around 2007 when the software development and IT operations communities raised concerns about the traditional software development model, where developers who wrote code worked apart from operations who deployed and supported the code. The term DevOps, a combination of the word's development and operations, reflects the process of integrating these disciplines into one, continuous process.

Working Of DevOps:



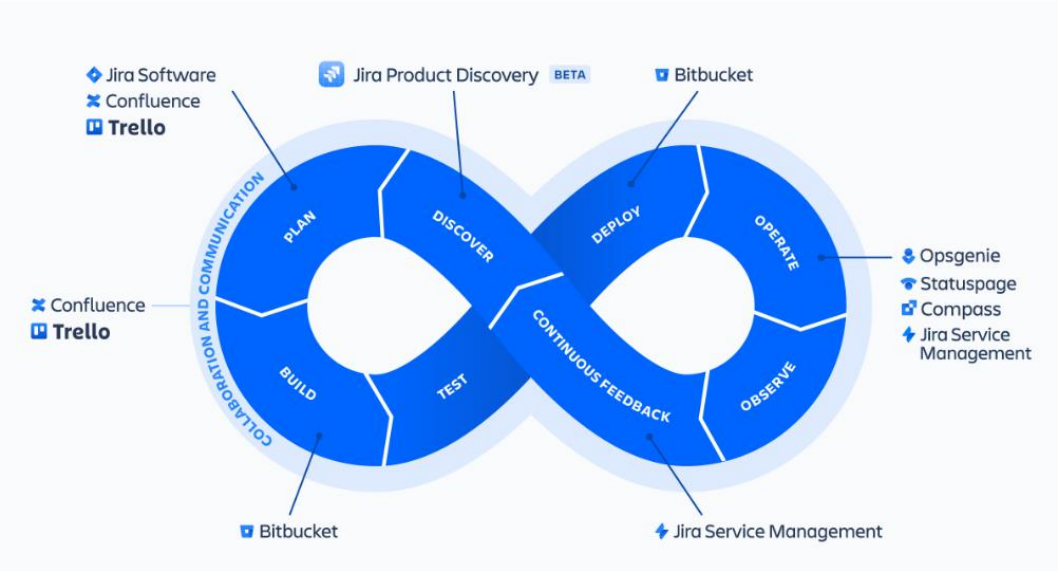
- A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, in order to increase the speed and quality of software deployment. It's a new way of working, a cultural shift, that has significant implications for teams and the organizations they work for.
- Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.
- DevOps teams use tools to automate and accelerate processes, which helps to increase reliability. A DevOps toolchain helps teams tackle important DevOps fundamentals including continuous integration, continuous delivery, automation, and collaboration.
- In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

DevOps Lifecycle:



1. **DISCOVER:**
   Building software is a team sport. In preparation for the upcoming sprint, teams must workshop to explore, organize, and prioritize ideas. Ideas must align to strategic goals and deliver customer impact. Agile can help guide DevOps teams.
2. **PLAN:**
   DevOps teams should adopt agile practices to improve speed and quality. Agile is an iterative approach to project management and software development that helps teams break work into smaller pieces to deliver incremental value.
3. **BUILD:**
   Git is a free and open-source version control system. It offers excellent support for branching, merging, and rewriting repository history, which has led to many innovative and powerful workflows and tools for the development build process.
4. **TEST:**
   Continuous integration (CI) allows multiple developers to contribute to a single shared repository. When code changes are merged, automated tests are run to ensure correctness before integration. Merging and testing code often help development teams gain reassurance in the quality and predictability of code once deployed.
5. **DEPLOY:**
   Continuous deployment (CD) allows teams to release features frequently into production in an automated fashion. Teams also have the option to deploy with feature flags, delivering new code to users steadily and methodically rather than all at once. This approach improves velocity, productivity, and sustainability of software development teams.
6. **OPERATE:**
   Manage the end-to-end delivery of IT services to customers. This includes the practices involved in design, implementation, configuration, deployment, and maintenance of all IT infrastructure that supports an organization's services.
7. **OBSERVE:**
   Quickly identify and resolve issues that impact product uptime, speed, and functionality. Automatically notify your team of changes, high-risk actions, or failures, so you can keep services on.
8. **CONTINEOUS FEEDBACK:**
   DevOps teams should evaluate each release and generate reports to improve future releases. By gathering continuous feedback, teams can improve their processes and incorporate customer feedback to improve the next release.

DevOps Tools:



- Confluence.
- Trello.
- JIRA Software
    - JIRA service management.
    - JIRA product discovery.
- Statuspage.
- Opsgenie.
- Compass.
- BitBucket.

Benefits of DevOps:

- **SPEED:**

  Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

- **RAPID DEPLOYMENT/DELIVERY:**

  Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage.

- **QUALITY and RELIABILITY:**

  Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

- **IMPROVED COLLABRATION:**

  Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

- **SCALE:**

  Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

- **SECURITY:**

  Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

Adopting DevOps:

1) **CONTINUOUS INTEGRATION:**

   Continuous integration is the practice of automating the integration of code changes into a software project. It allows developers to frequently merge code changes into a central repository where builds and tests are executed. This helps DevOps teams address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

2) **CONTINUOUS DELIVERY:**

   Continuous delivery expands upon continuous integration by automatically deploying code changes to a testing/production environment. It follows a continuous delivery pipeline, where automated builds, tests, and deployments are orchestrated as one release workflow.

3) **SITUATIONAL AWARENESS:**

   It is vital for every member of the organization to have access to the data they need to do their job as effectively and quickly as possible. Team members need to be alerted of failures in the deployment pipeline — whether systemic or due to failed tests — and receive timely updates on the health and performance of applications running in production. Metrics, logs, traces, monitoring, and alerts are all essential sources of feedback teams need to inform their work.

4) **AUTOMATION:**

   Automation is one of the most important DevOps practices because it enables teams to move much more quickly through the process of developing and deploying high-quality software. With automation the simple act of pushing code changes to a source code repository can trigger a build, test, and deployment process that significantly reduces the time these steps take.

5) INFRASTRUCTURE AS CODE:

Whether your organization has an on-premise data center or is completely in the cloud, having the ability to quickly and consistently provision, configure, and manage infrastructure is key to successful DevOps adoption. Infrastructure as Code (IaC) goes beyond simply scripting infrastructure configuration to treating your infrastructure definitions as actual code: using source control, code reviews, tests, etc.

6) MICROSERVICES:

Microservices is an architectural technique where an application is built as a collection of smaller services that can be deployed and operated independently from each other. Each service has its own processes and communicates with other services through an interface. This separation of concerns and decoupled independent function allows for DevOps practices like continuous delivery and continuous integration.

7) MONITORING:

DevOps teams monitor the entire development lifecycle — from planning, development, integration and testing, deployment, and operations. This allows teams to respond to any degradation in the customer experience, quickly and automatically. More importantly, it allows teams to "shift left" to earlier stages in development and minimize broken production changes.

Agile:

- Agile methodologies are immensely popular in the software industry since they empower teams to be inherently flexible, well-organized, and capable of responding to change. DevOps is a cultural shift that fosters collaboration between those who build and maintain software. When used together, agile and DevOps result in high efficiency and reliability.
- Organizations that do DevOps well are places where experimentation and some amount of risk-taking are encouraged. Where thinking outside the box is the norm, and failure is understood to be a natural part of learning and improving.
- A DevOps culture is where teams embrace new ways of working that involve greater collaboration and communication. It's an alignment of people, processes, and tools toward a more unified customer focus. Multidisciplinary teams take accountability for the entire lifecycle of a product.

Result:

Thus, the study about the purpose of cloud computing DevOps has been known successfully.